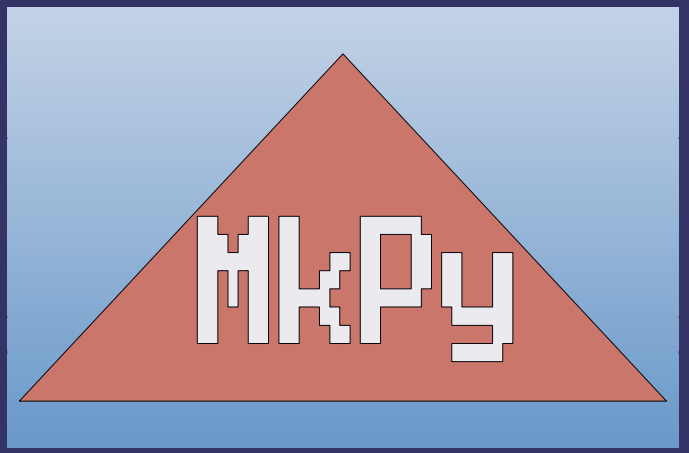
'Let them evolve!'
Auto scheduling using Python and Biology

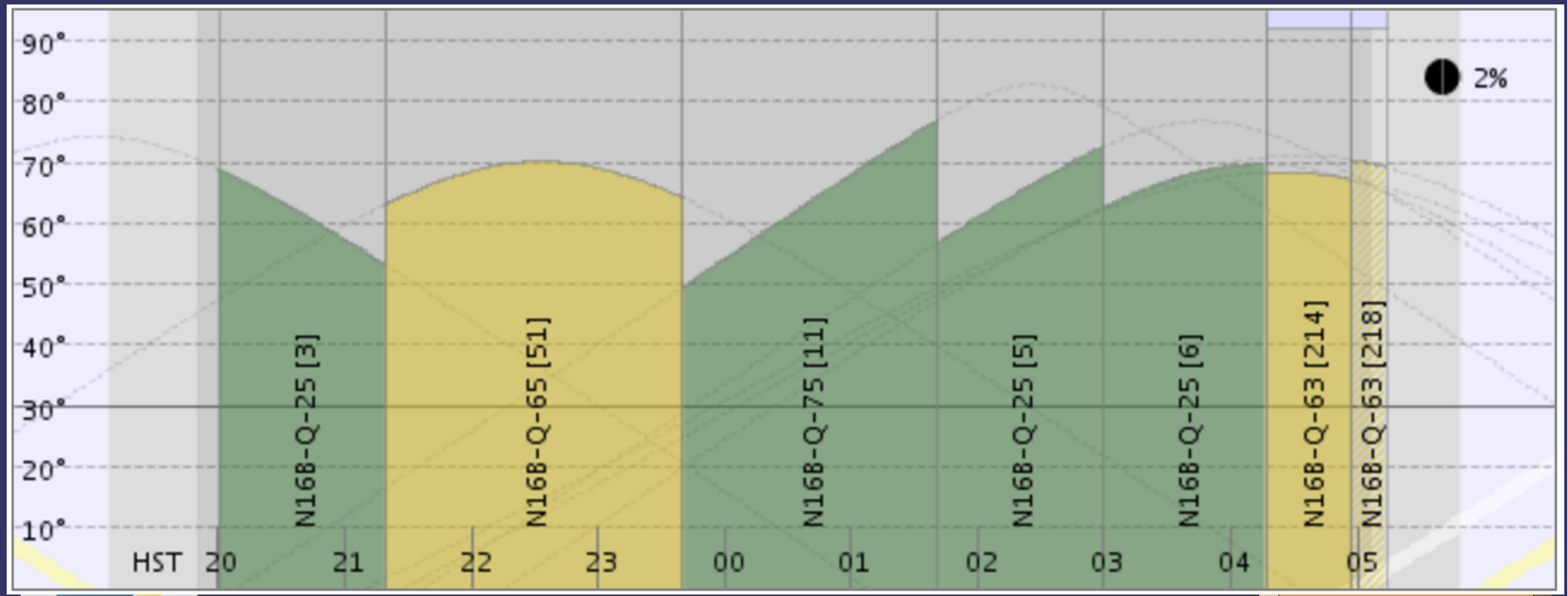
Michael Pohlen



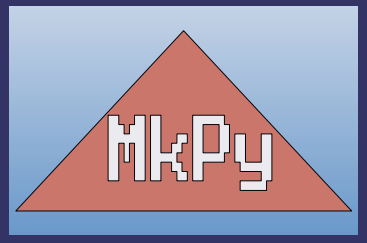
MkPy Workshop
8.-9. November 2017

What is scheduling?

- Distribution of requested/desired observation over night

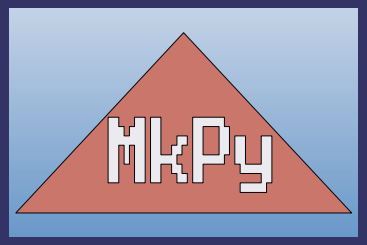


Example: Gemini observatory



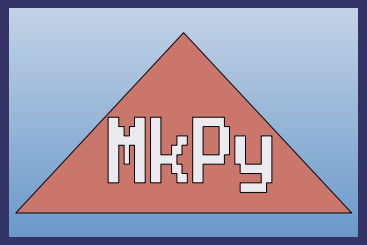
Why is it an issue?

- Classical observer
  - Small amount of observations
  - Knows priority
  - Flexible, i.e. can react to weather, SN, …
    ==> rel. simply done by hand especially using QPT, or skycalc
- Queue observatory
  - Lots of observations
  - Many constraints (conditions, science bands, partner shares,..)
    ==> tough task often done by dedicated FTE
- Robotic telescopes
  - Small number of staff
  - React to change in conditions/importance automatically
    ==> Unavoidable and very important



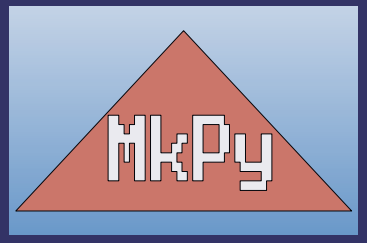
What did I look into?

- Looking for student project at UHH summer course (Software Systems for Astronomy)
- Al Conrad (UH) suggest to look into automatic scheduling: Python's object orientated setup + a genetic algorithm
- Based on idea presented by Petr Kubánek in
  "Genetic Algorithm for Robotic Telescope Scheduling"
  2009 Master Thesis (University Granada, Spain)
- Then a planned extension to RTS2
  (open source package for autonomous observatories)



Multi Objective Optimisation

- Using evolutionary/genetic algorithm: NSGA-II (Non-dominated Sorting Genetic Algorithm II, Deb et al. 2002)
  - 1) Initial random population of schedules
  - 2) Rank schedules (Non-dominated sort, Pareto front)
  - 3) Evolve population:
    - Crossover: best parents mate and have two children (tournament selection)
      - child 1 gets most of observation from p1, rest from p2
      - child 2 gets most of observation from p2, rest from p1
    - Mutation: random observation replaced
  - 4) Add children to population, rank, and cut back to original size
  - 5) Start over at number 3)



Python Objects/Definitions

site
- name
- lat
- lon
-elevation

target
- ra
-dec
- tobs
- maxAlt
- getMidHa()

obsPool
- all observation

schedule
(list per night)
- addObs()
- mergeSched()
- pushLeft()
- swap()
- fillObs()
- getIdList()
-getAmRange()
- plotSched()
- getFit()

population
(of schedules)
- addSched()
- extend()
- plotFit()

evolution
- evolve
- createInitial

NSGA-II
- tournament()
- createChildren()
- calcRank()
-calcCrowdingDist()

getRandomRaDec

calcTrans

twilight

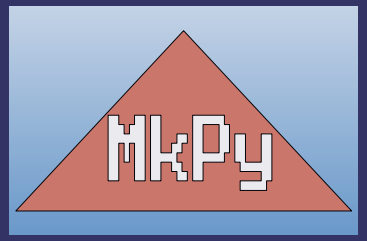
setObserver

convUt2Lt

convLt2Ut

calcAm

...



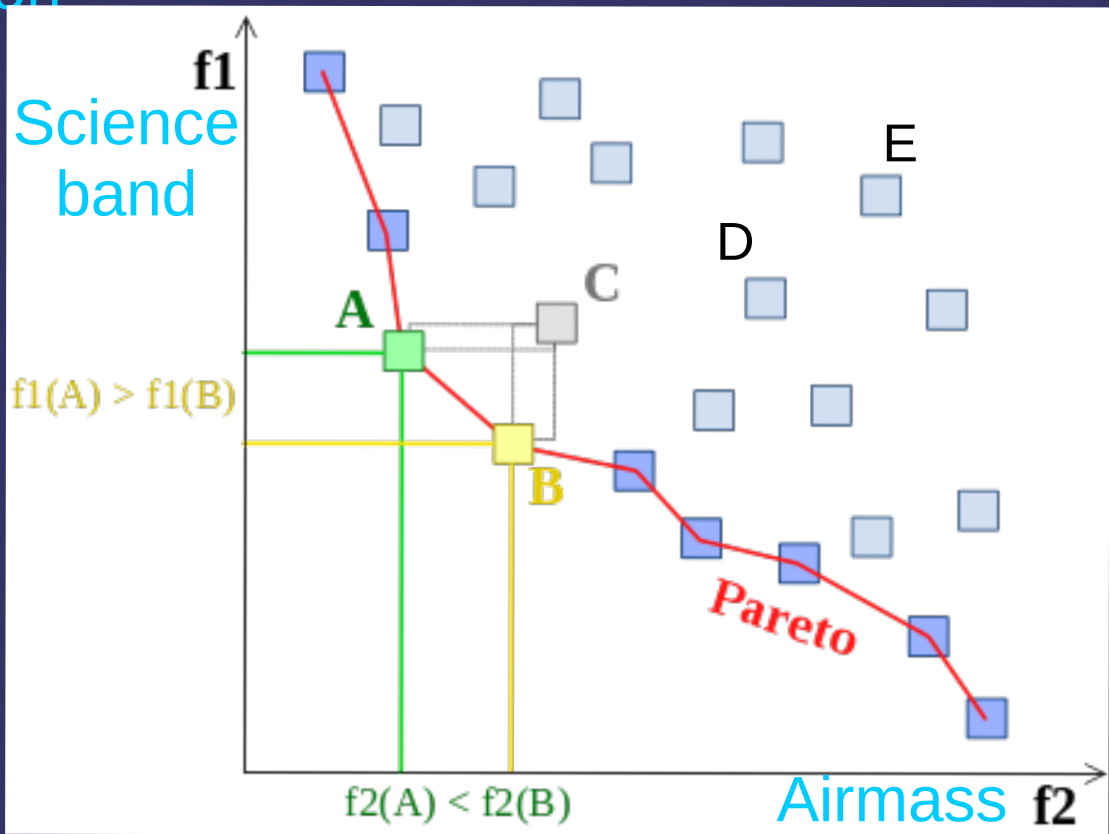
Algorithm details

- Objectives (minimising)
  - Science rank/band (e.g. @Gemini: 1-4)
    == band * Texp
  - Airmass (*the lower the better)*
    == midHA*(maxAlt-30)

Algorithm details

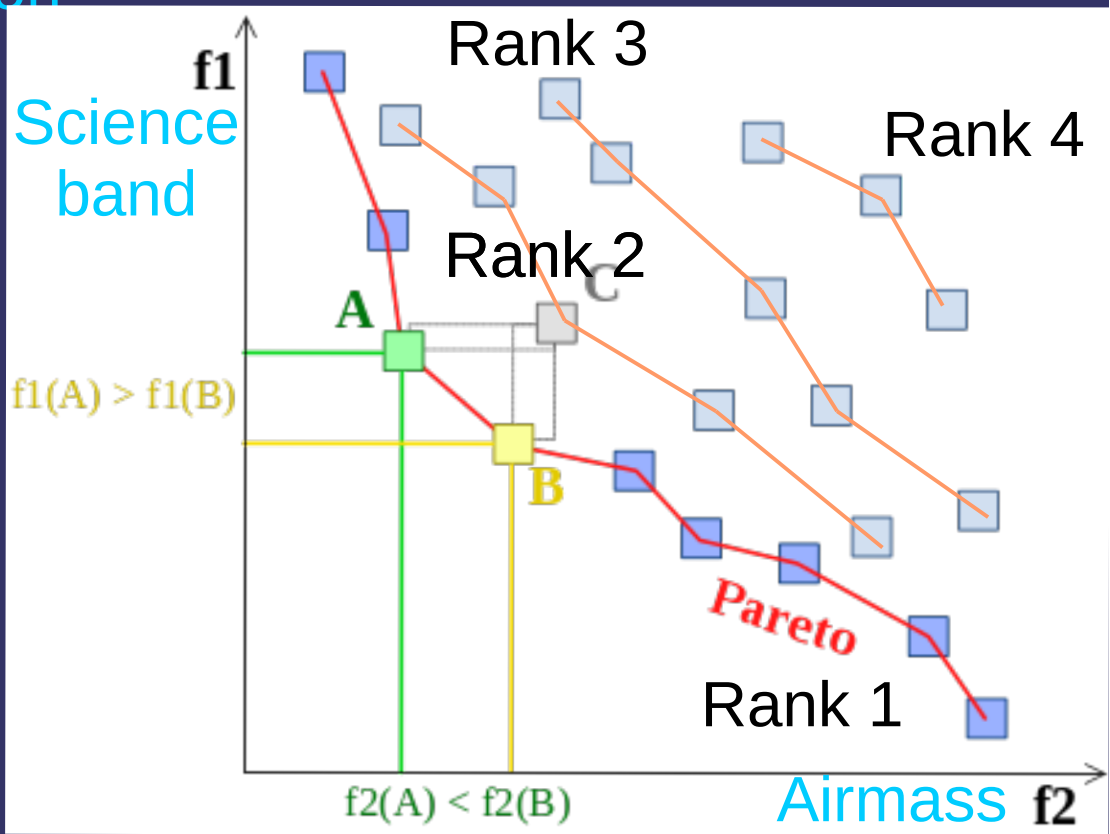
- Objectives (minimising)
  - Science rank/band (e.g. @Gemini: 1-4) == band * Texp
  - Airmass (*the lower the better)* == midHA*(maxAlt-30)

- Multi-objetive optimization
  - No single solution
  - Pareto optimal solutions (Pareto front)



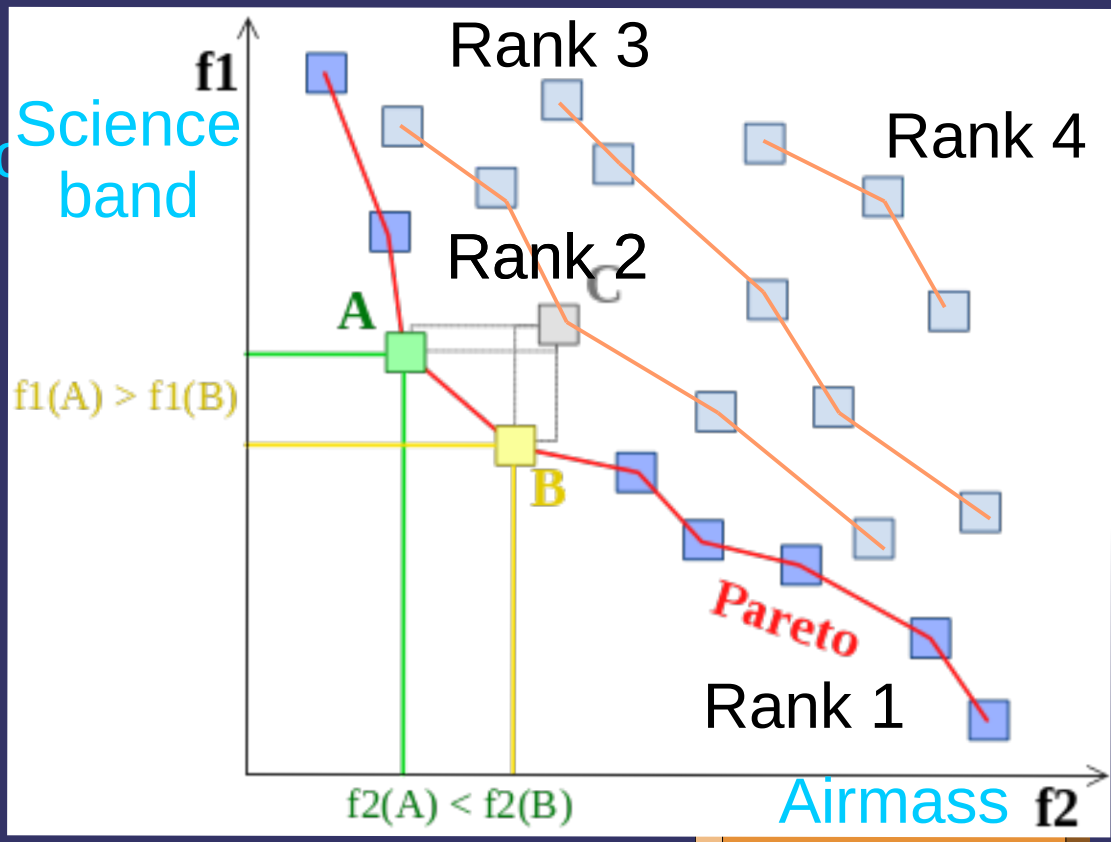
Algorithm details

- Objectives (minimising)
  - Science rank/band (e.g. @Gemini: 1-4)
    == band * Texp
  - Airmass (*the lower the better)*
    == midHA*(maxAlt-30)
- Multi-objetive optimization
  - No single solution
  - Pareto optimal solutions
    (Pareto front)
- Pareto based ranking
  - No weights needed
  - No combined fit needed

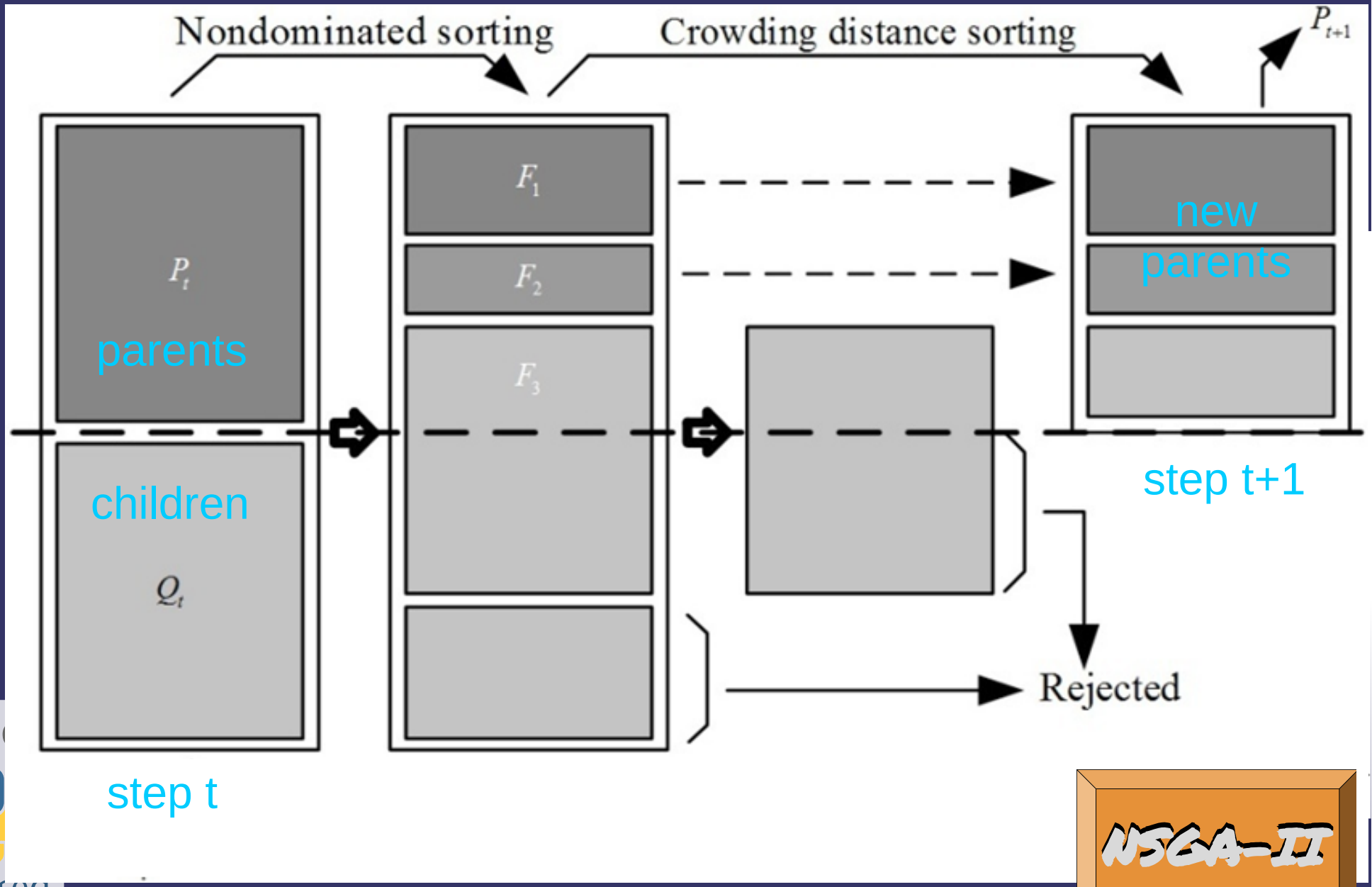


Algorithm details

- Objectives (minimising)
  - Science rank/band (e.g. @Gemini: 1-4)
    == band * Texp
  - Airmass (*the lower the better)*
    == midHA*(maxAlt-30)
- Multi-objetive optimizatio
  - No single solution
  - Pareto optimal solutions
    (Pareto front)
- Pareto based ranking
  - No weights needed
  - No combined fit needed
- Crowding distance
  - crowding distance to preserve ‘diversity’

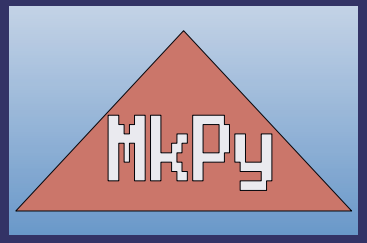


Algorithm details



powered

NSGA-II

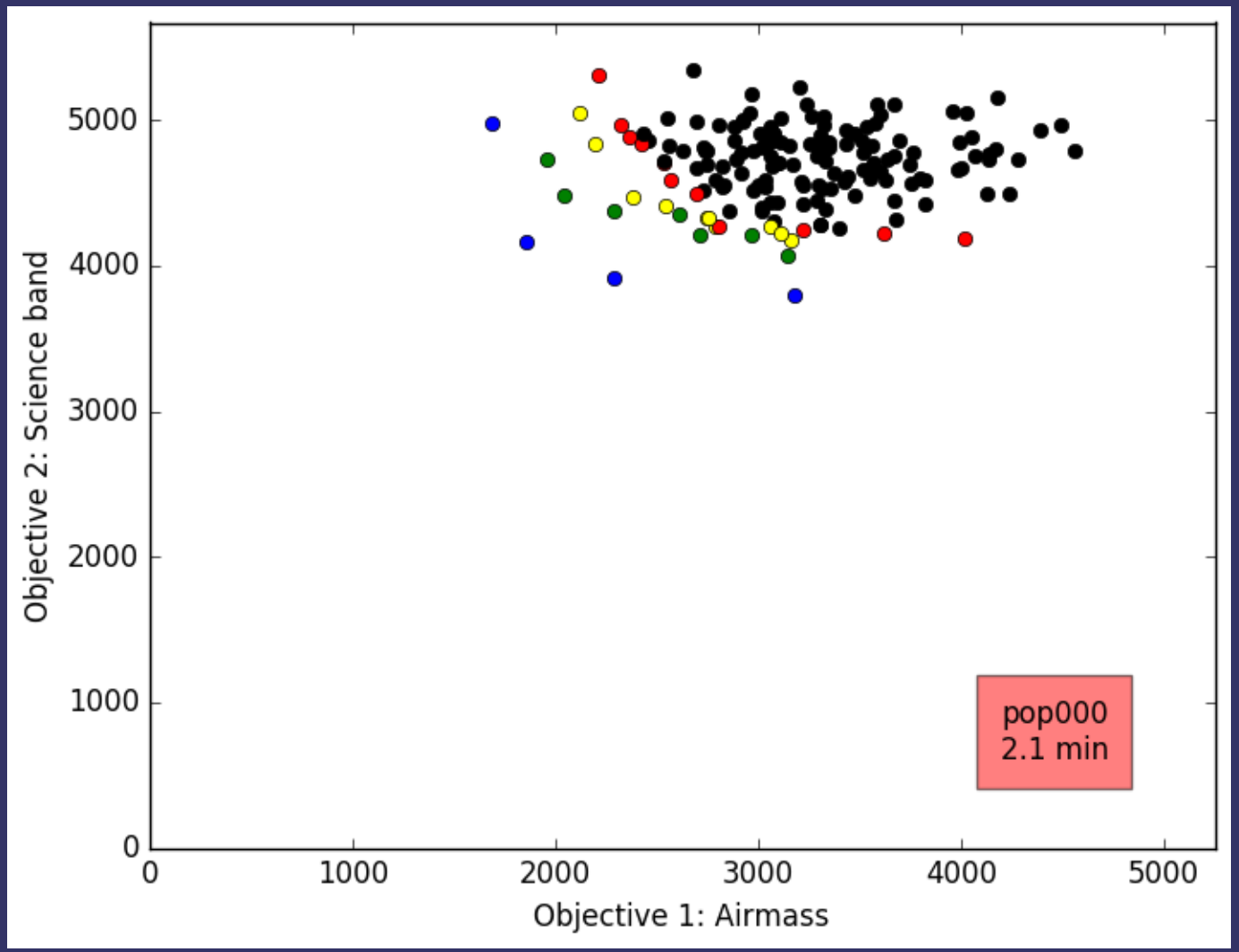


Scheduling specifies

- Merit defined as sum over nights
- Where to cut schedules?
  - By numObs, crossoverpoint 75%
- Crossover will leave gaps
  - Observation from other parent will not match gap 1-1
  - Some observations might be already schedules on other nights
- Repair schedule:
  - Shift new observations left
  - Fill gaps with random (free) observation from obsPool
- Swap observations:
  - Often a simple swap is beneficial

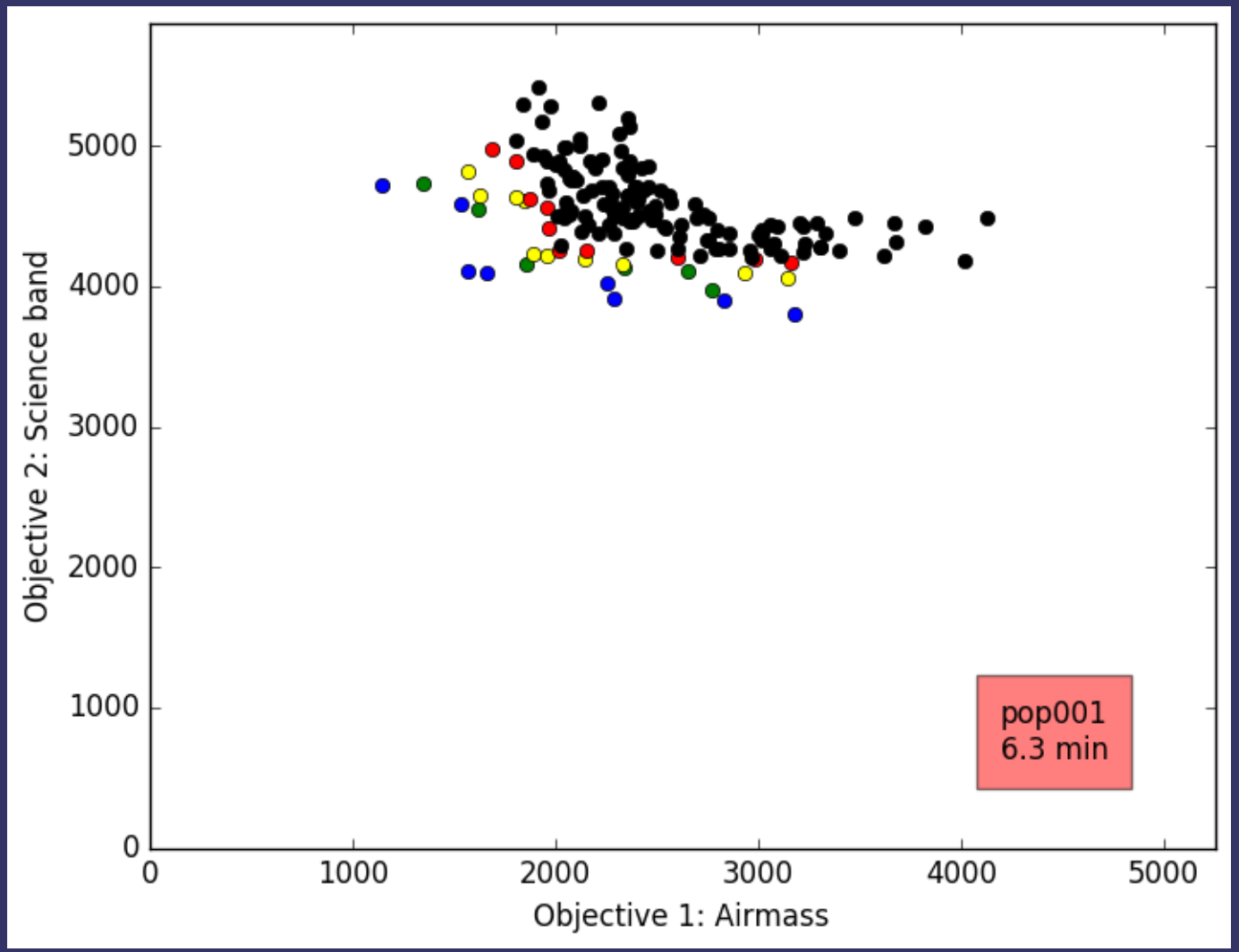
Example run

- 100 populations with 160 schedules each for 4 nights



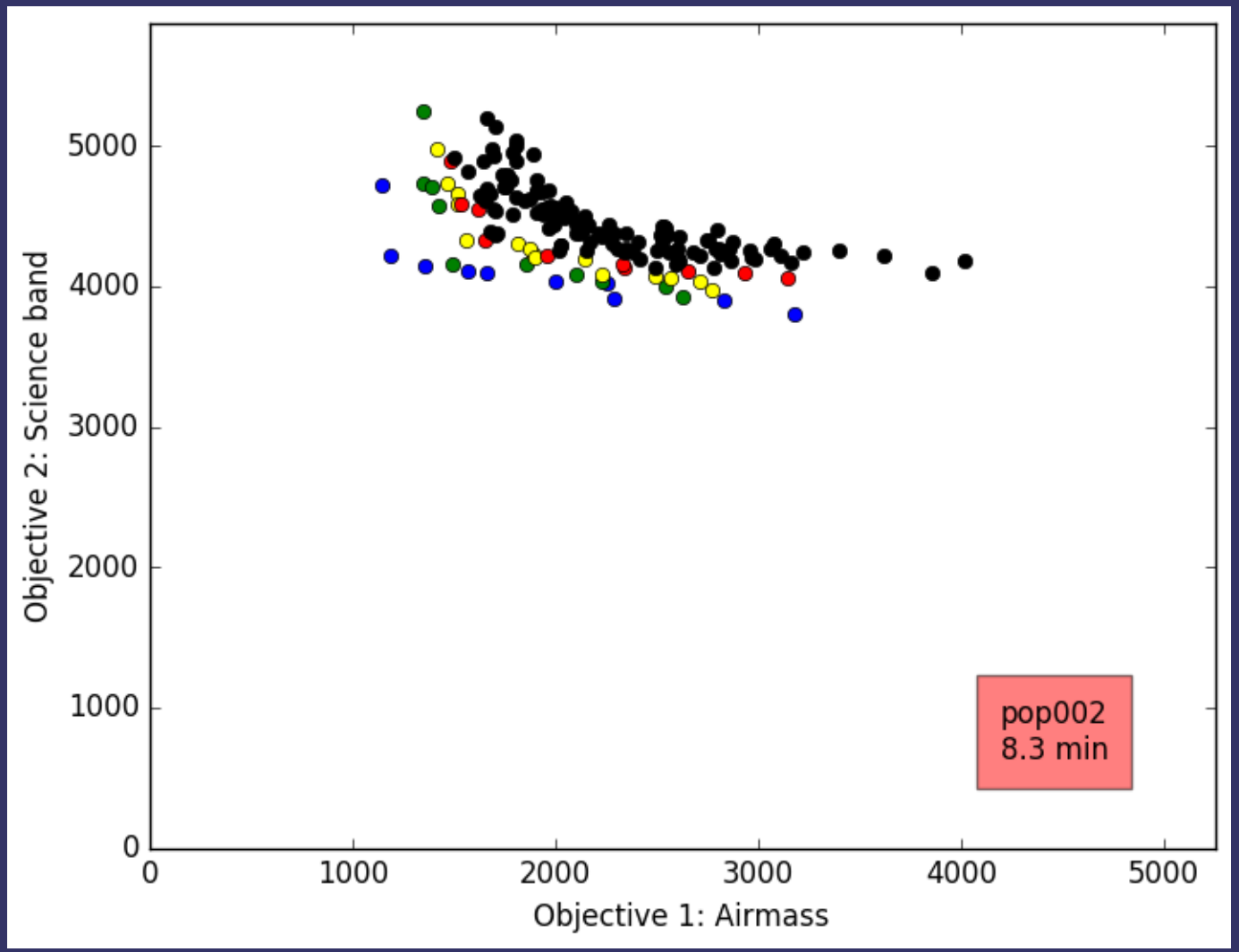
Example run

- 100 populations with 160 schedules each for 4 nights



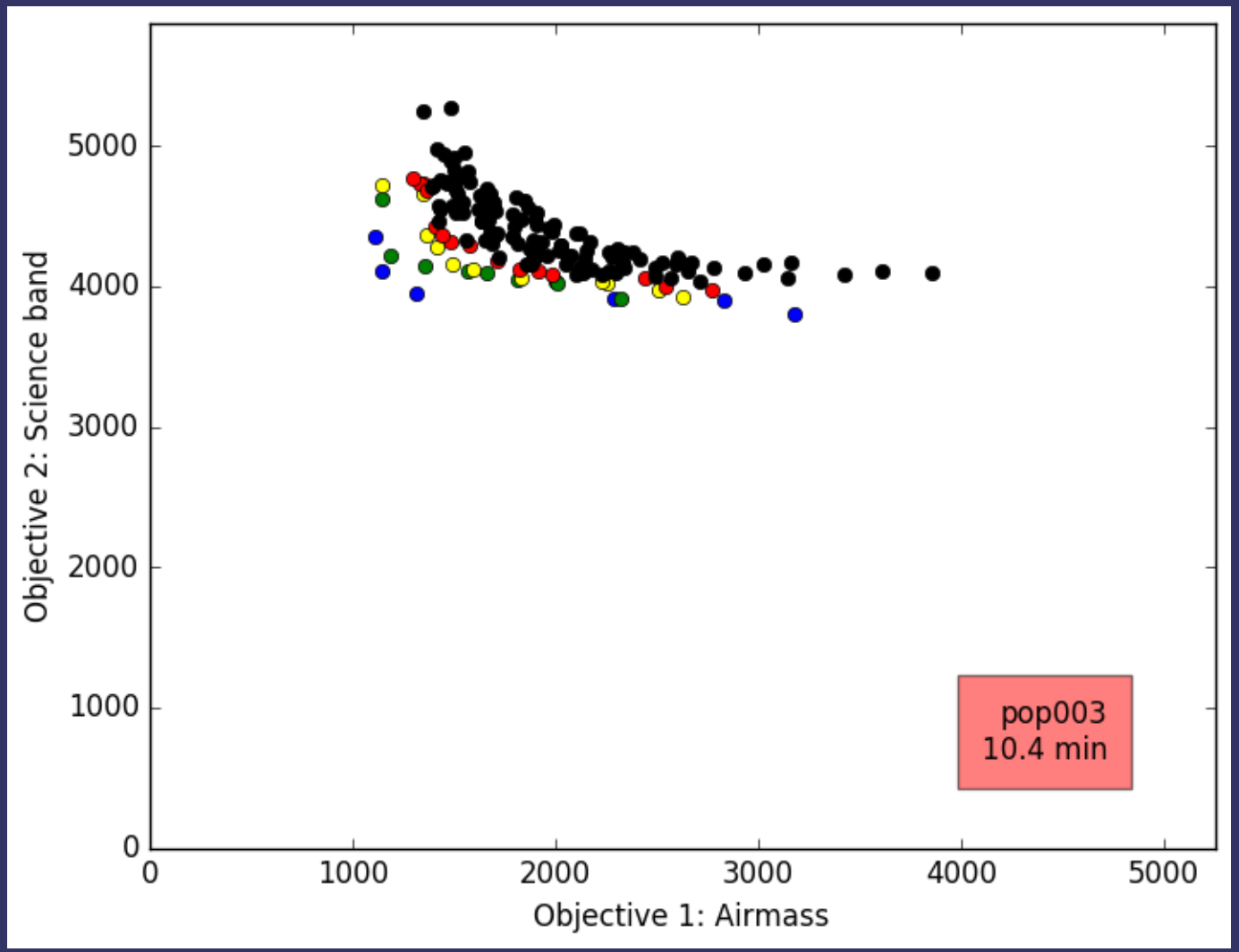
Example run

- 100 populations with 160 schedules each for 4 nights



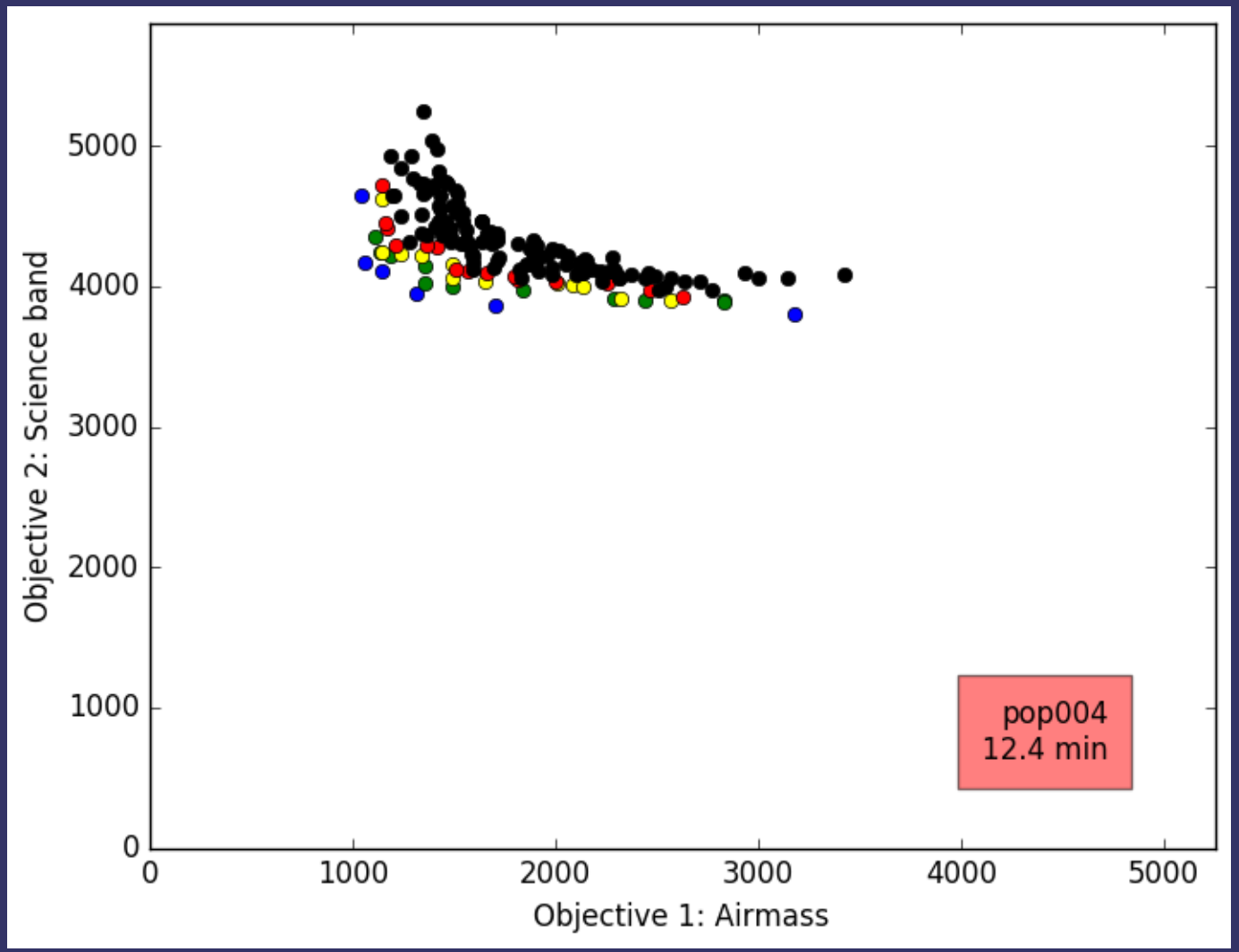
Example run

- 100 populations with 160 schedules each for 4 nights



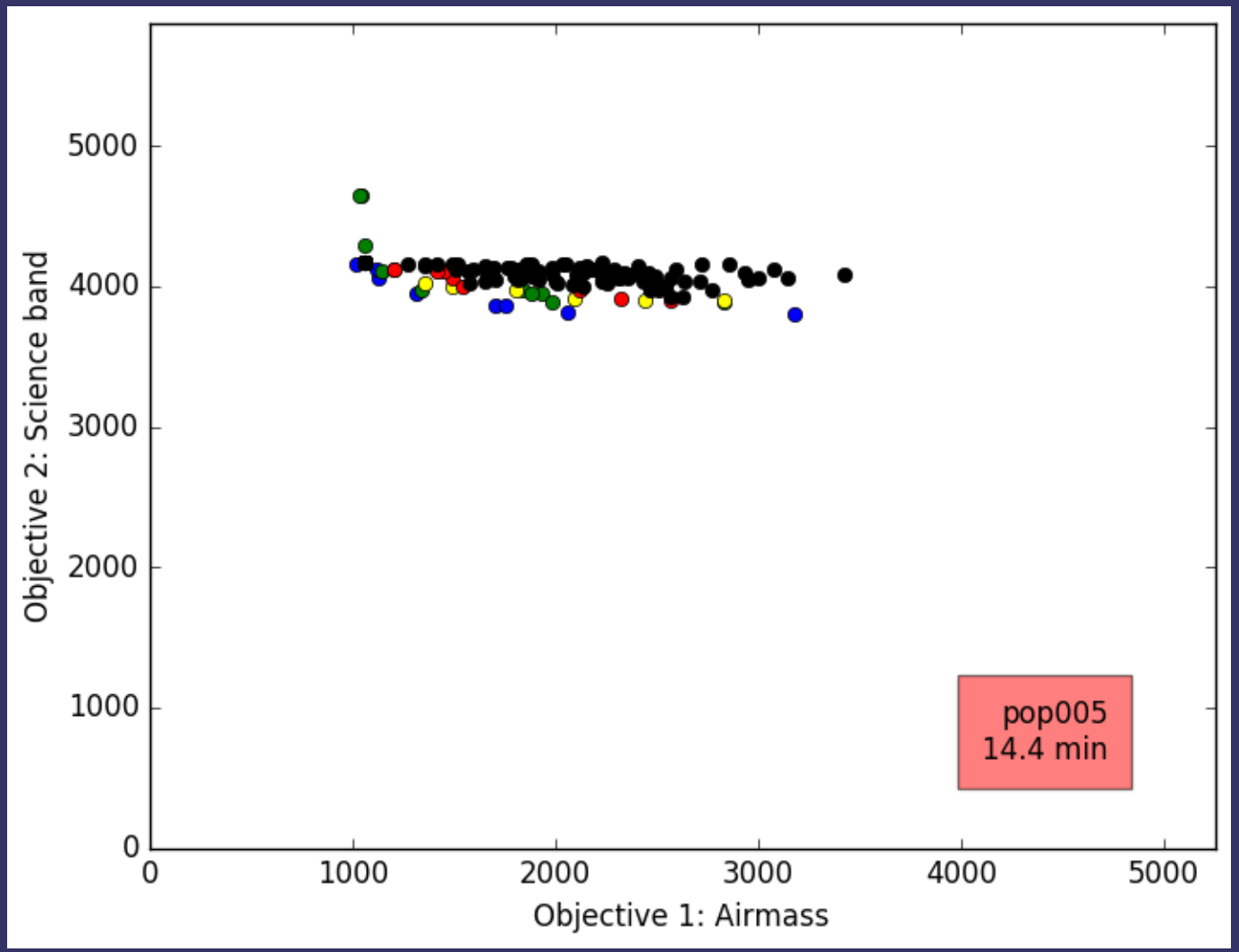
Example run

- 100 populations with 160 schedules each for 4 nights



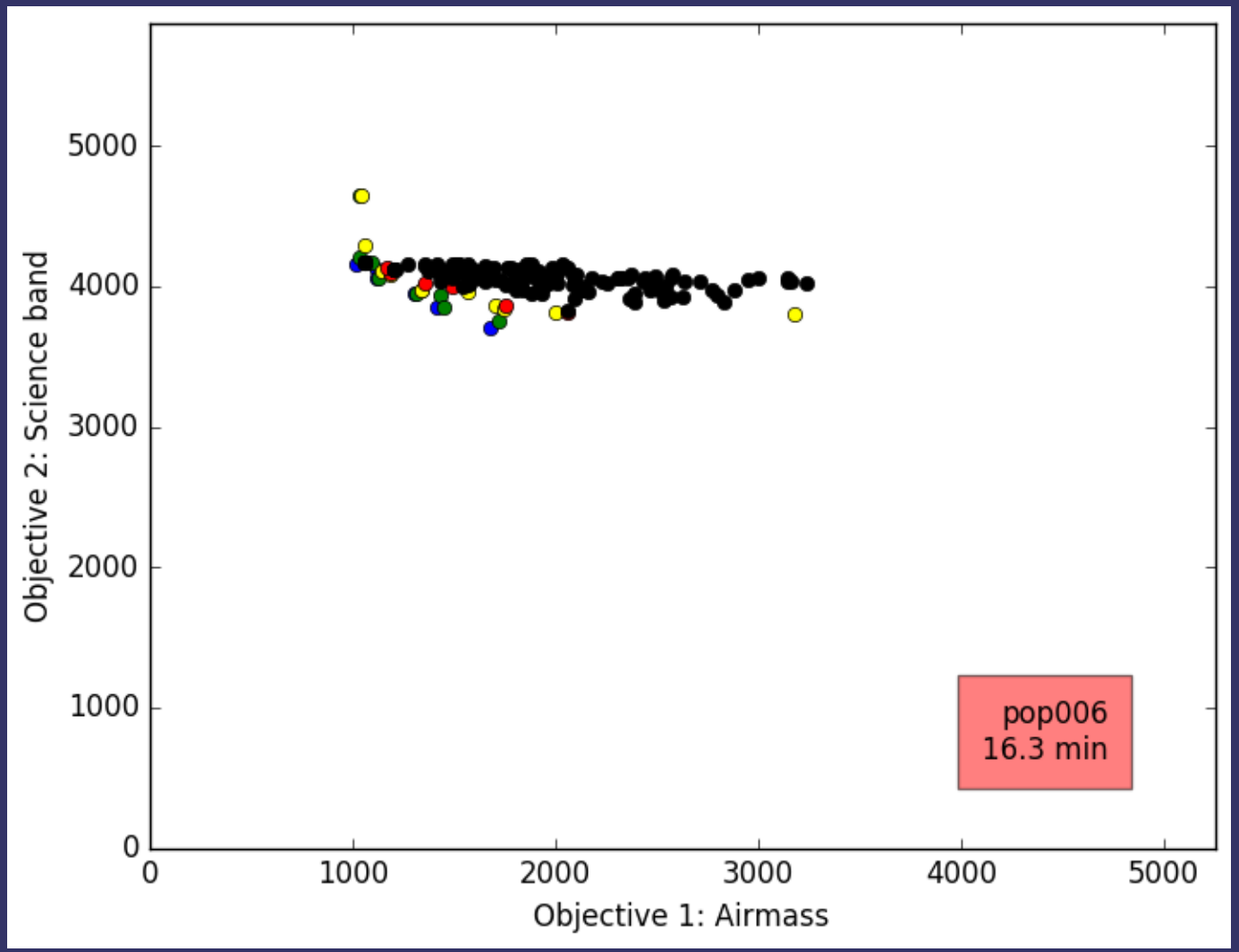
Example run

- 100 populations with 160 schedules each for 4 nights



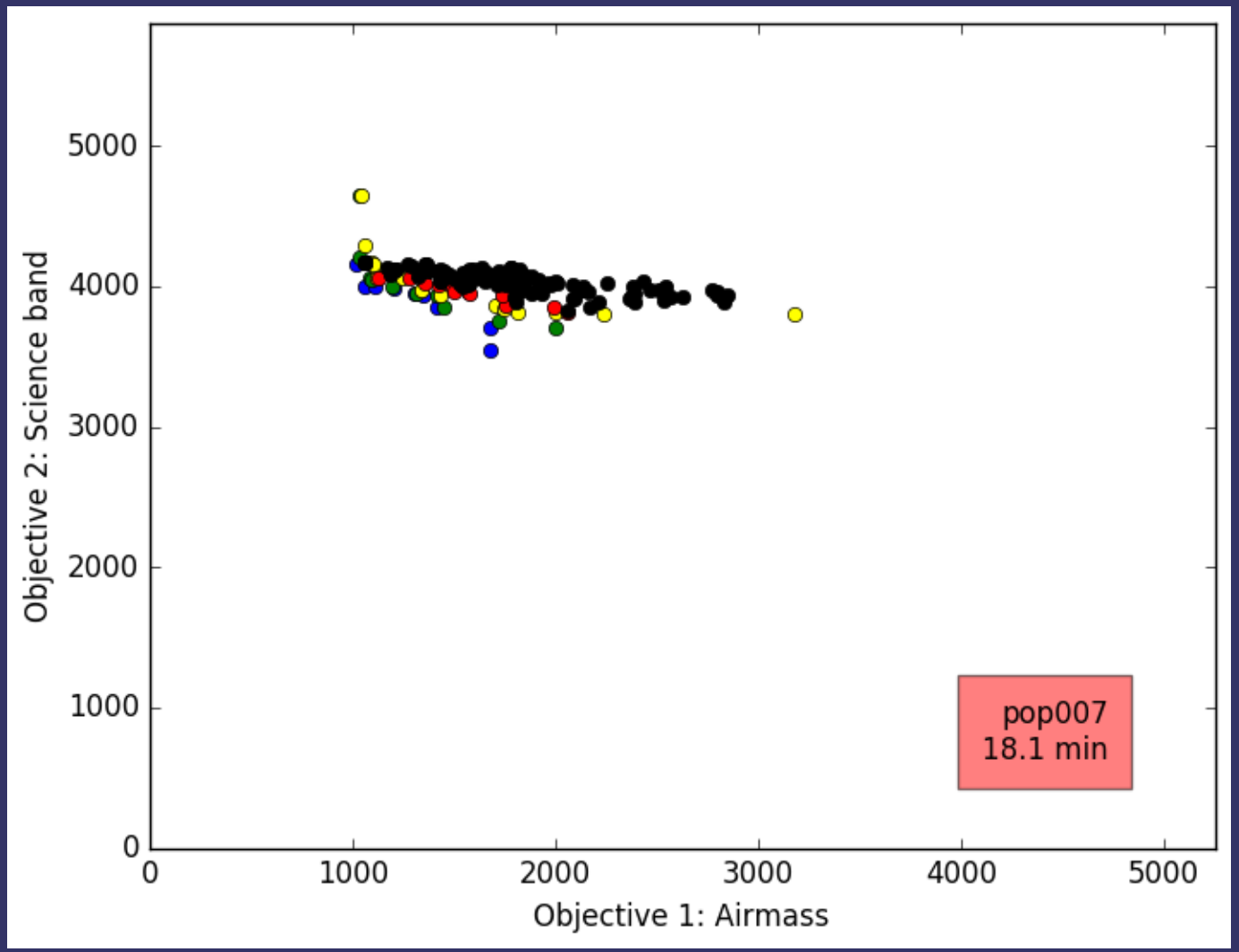
Example run

- 100 populations with 160 schedules each for 4 nights



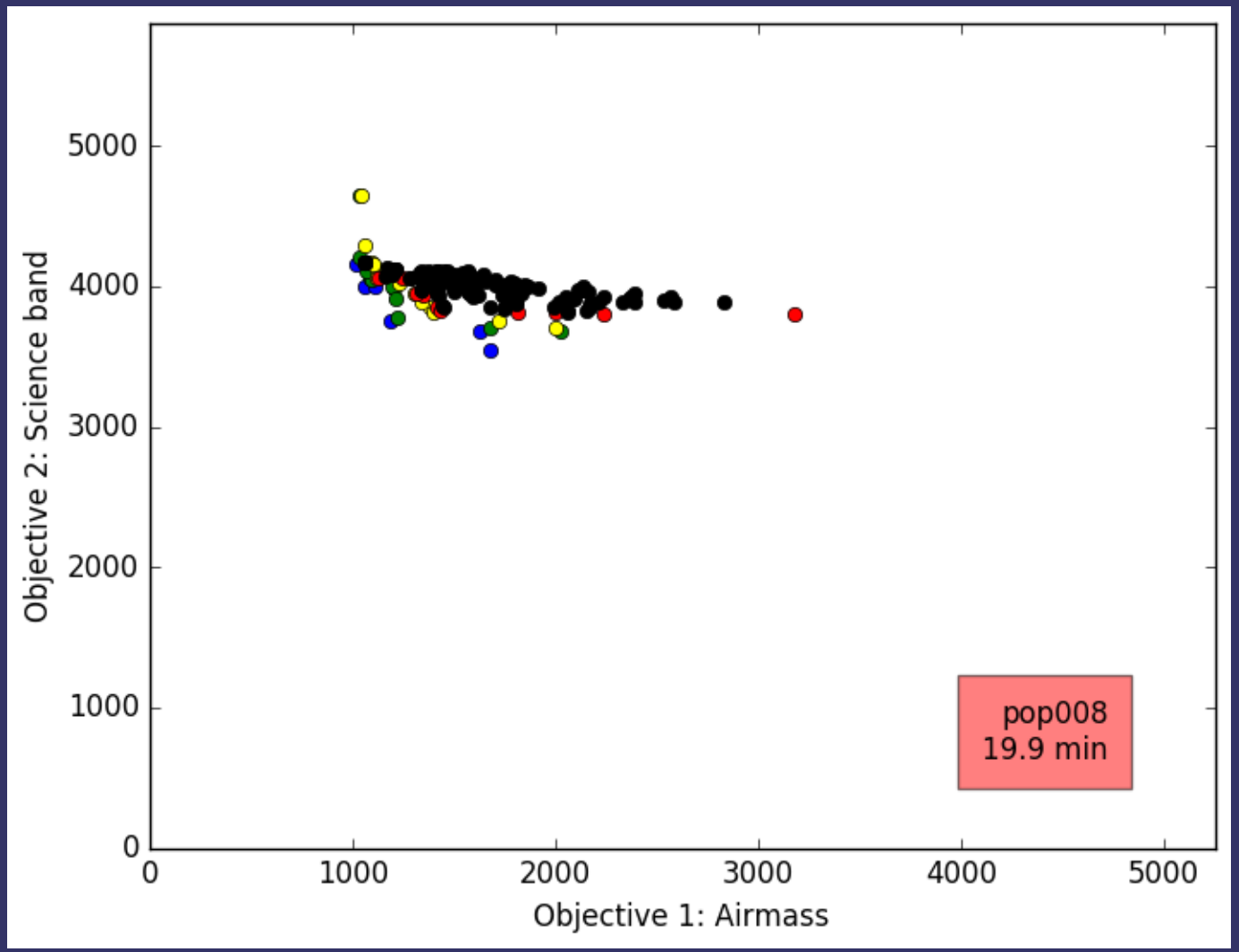
Example run

- 100 populations with 160 schedules each for 4 nights



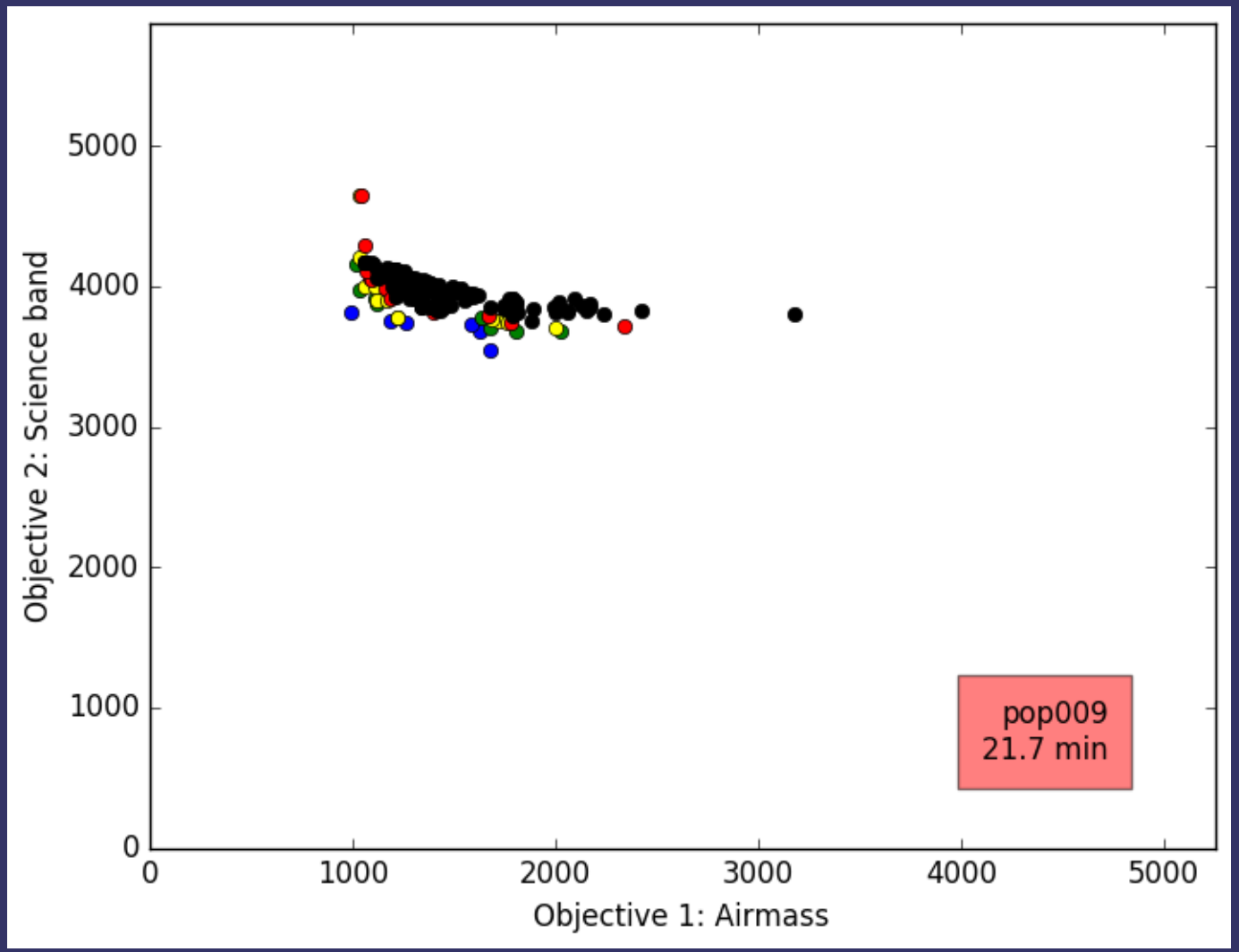
Example run

- 100 populations with 160 schedules each for 4 nights



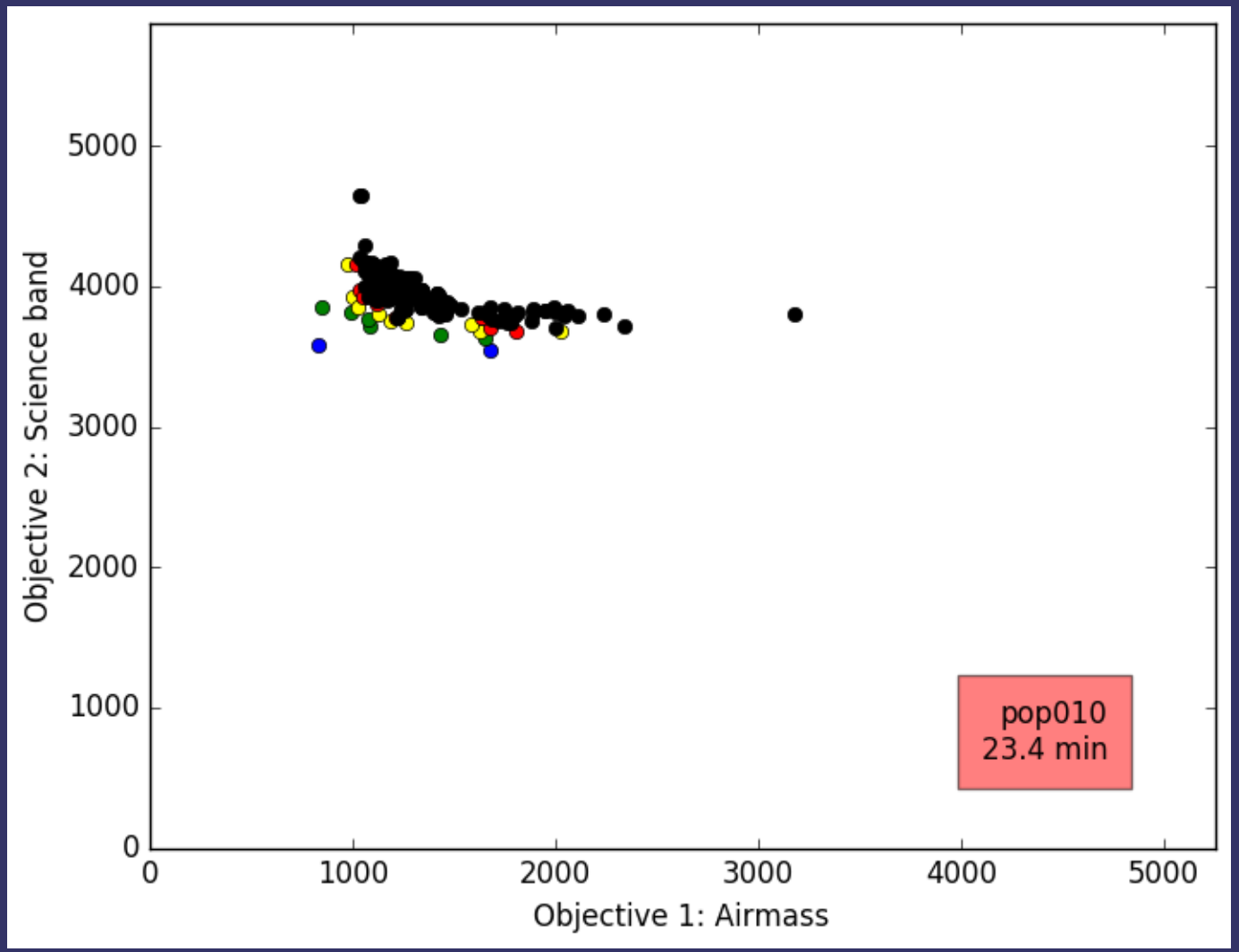
Example run

- 100 populations with 160 schedules each for 4 nights



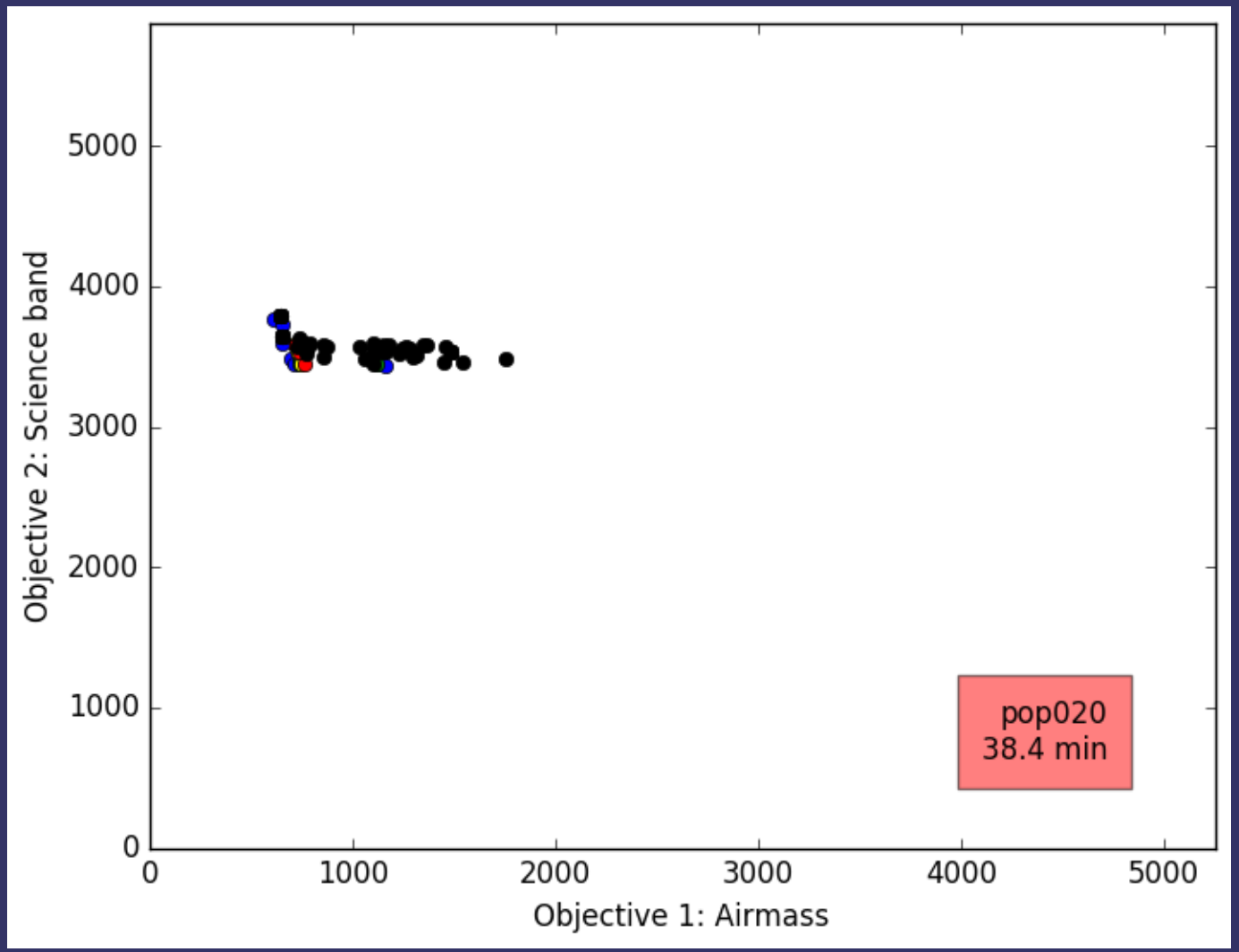
Example run

- 100 populations with 160 schedules each for 4 nights



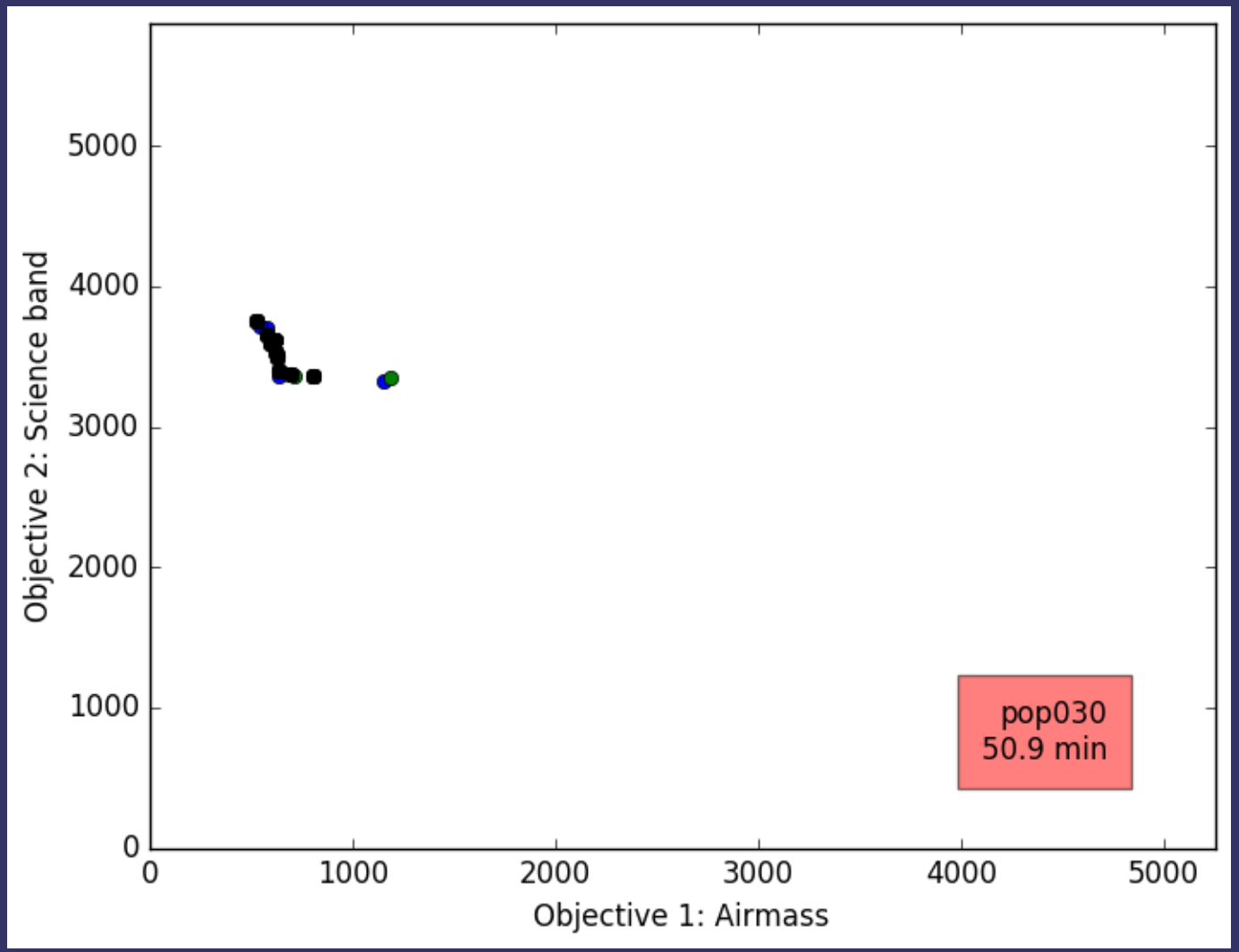
Example run

- 100 populations with 160 schedules each for 4 nights



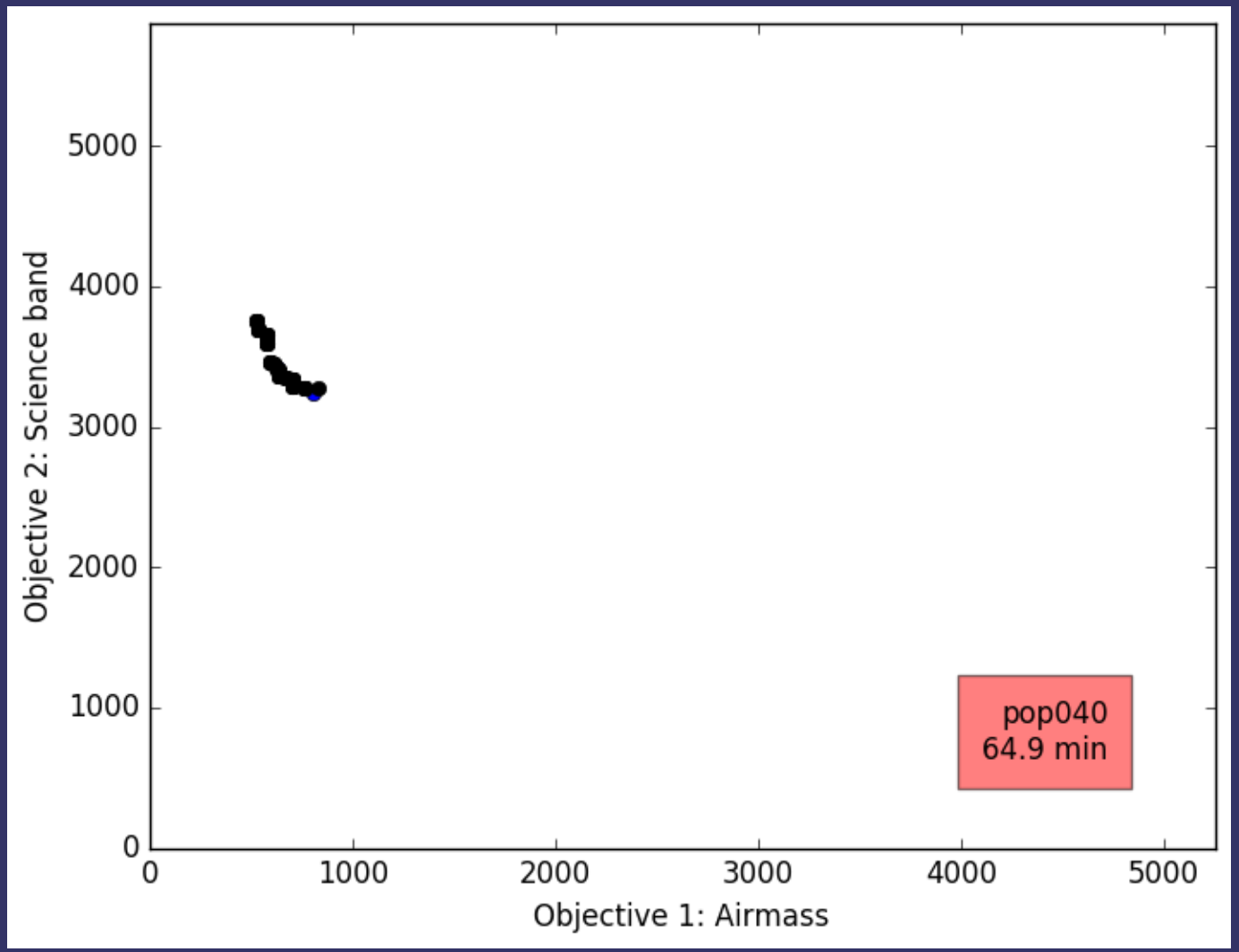
Example run

- 100 populations with 160 schedules each for 4 nights



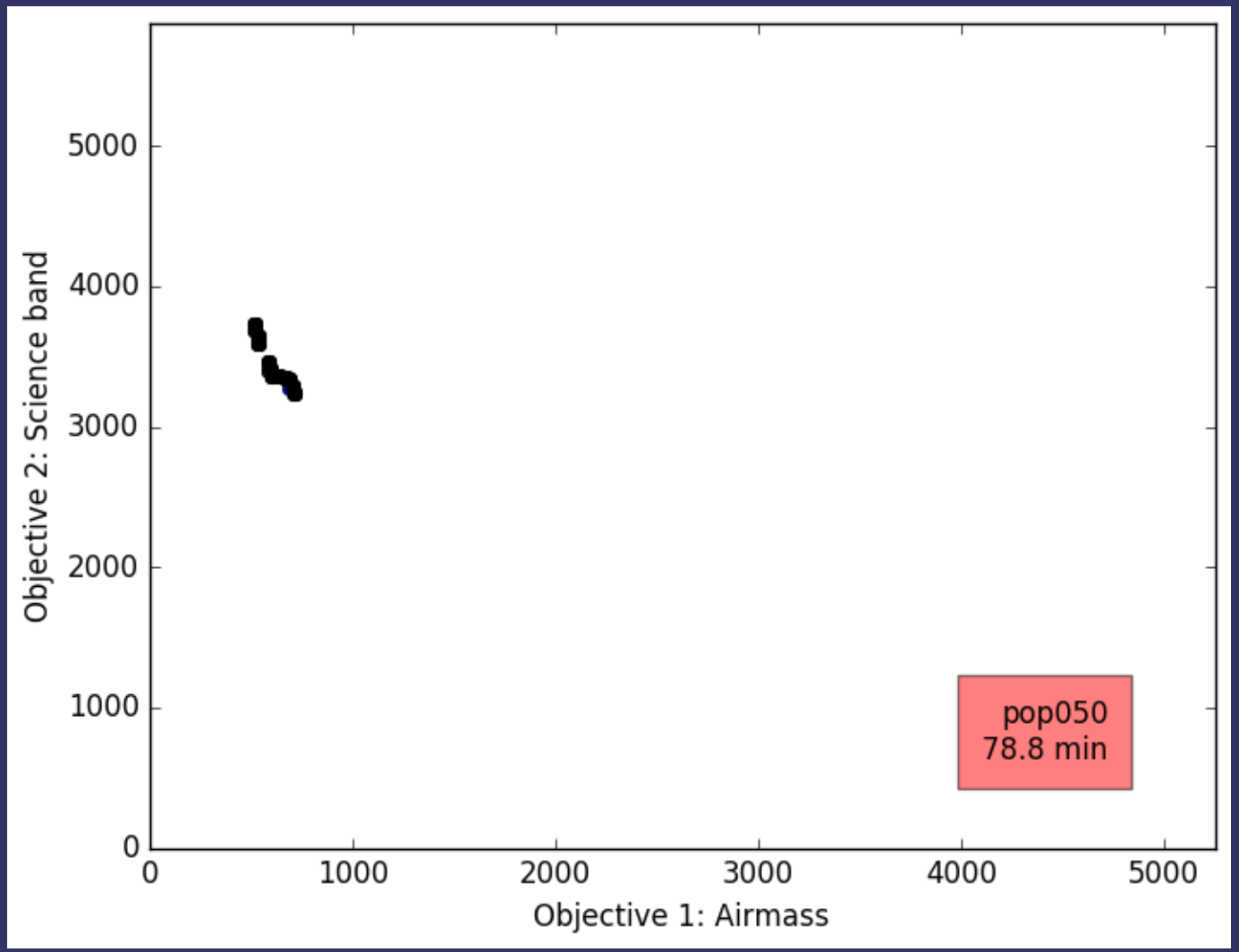
Example run

- 100 populations with 160 schedules each for 4 nights



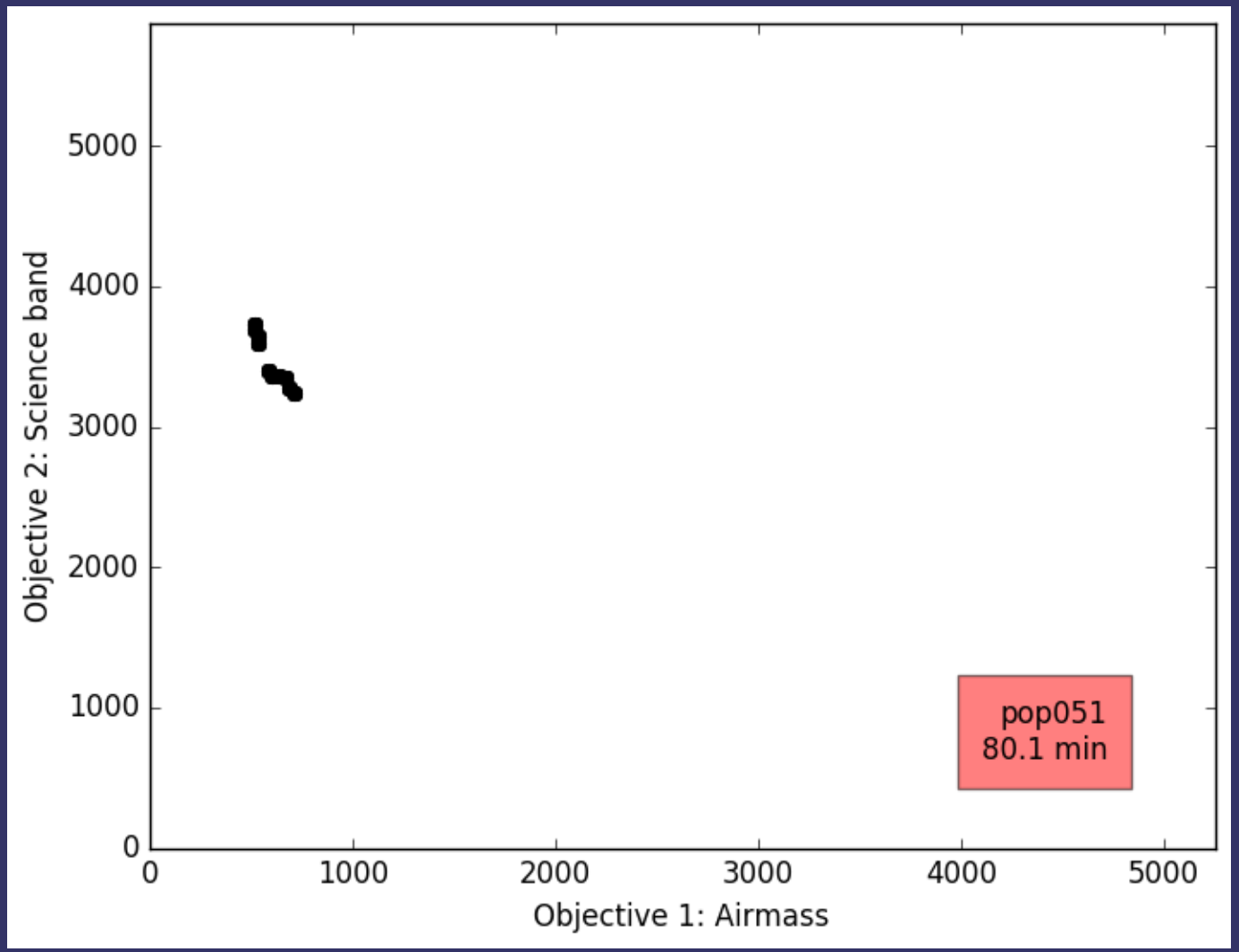
Example run

- 100 populations with 160 schedules each for 4 nights



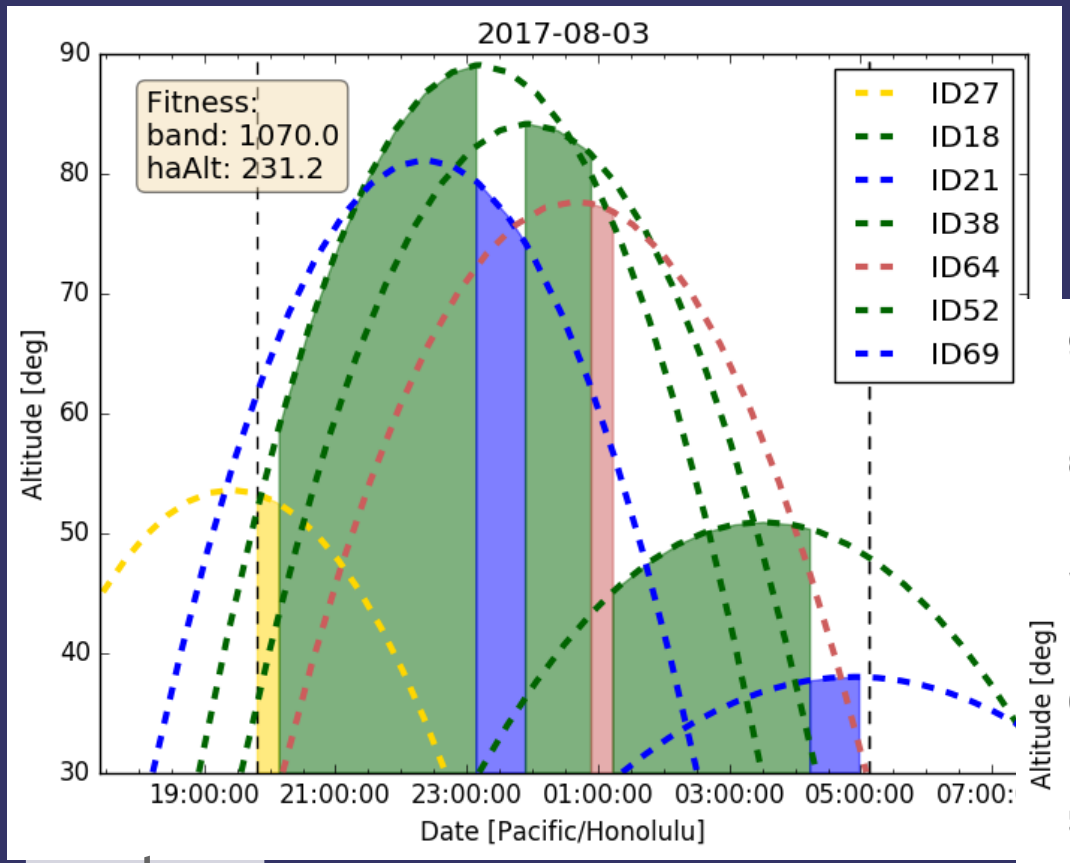
Example run

- 100 populations with 160 schedules each for 4 nights



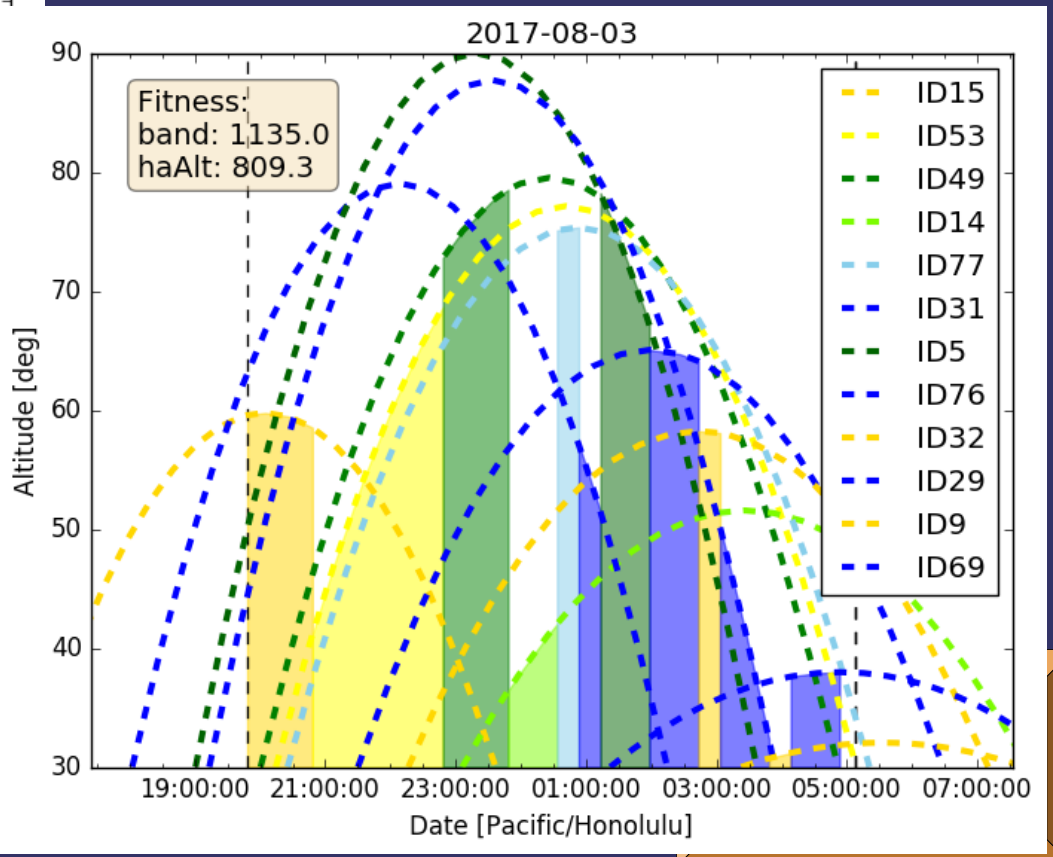
Schedules

- Initial random schedules



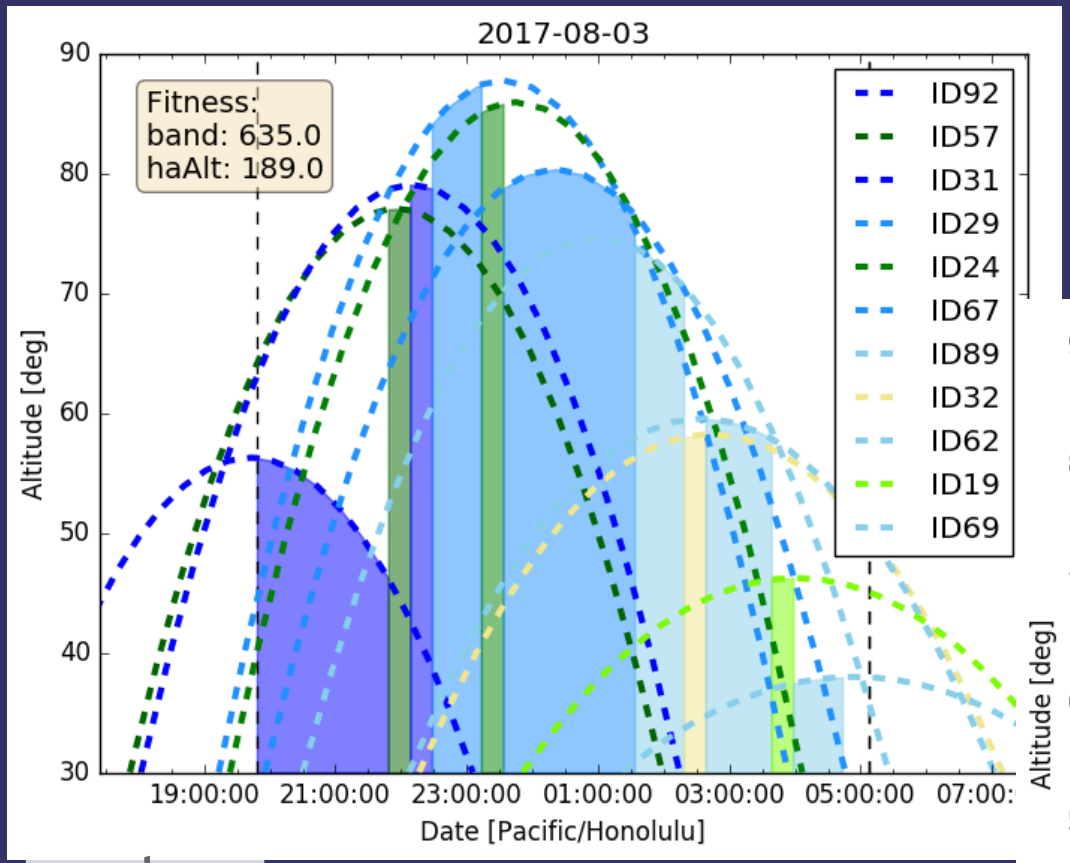
OK-ish

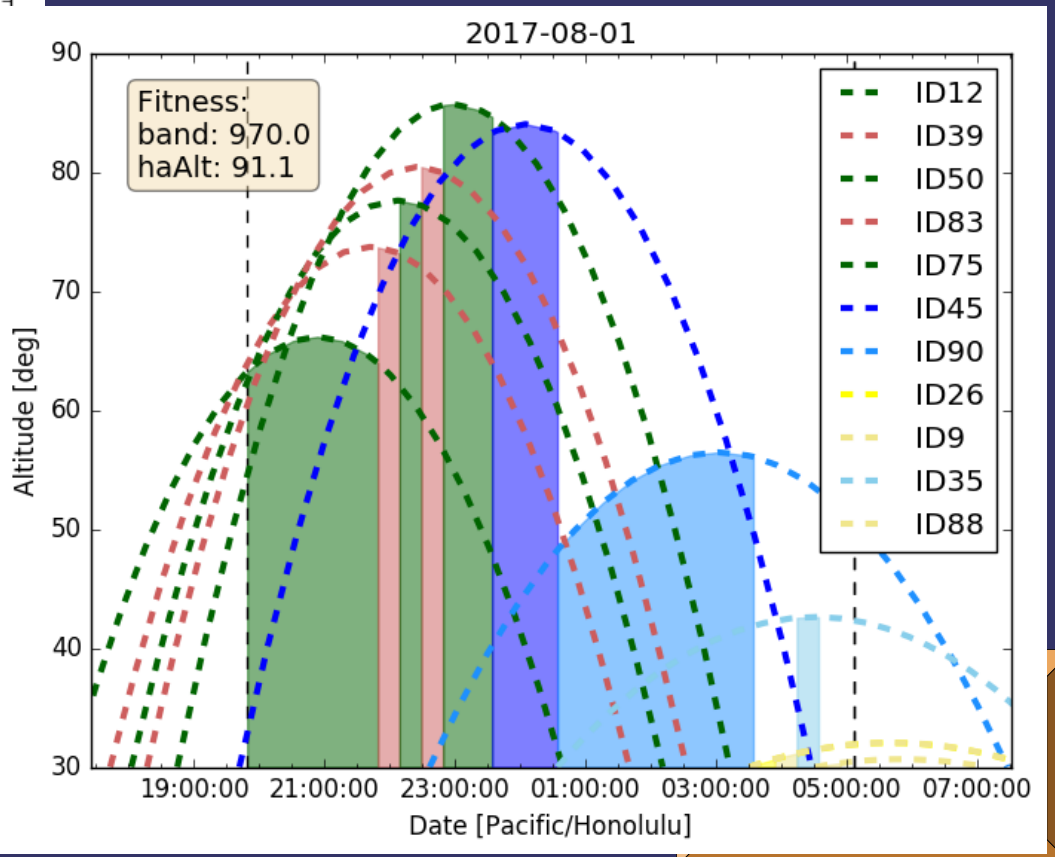
Bad

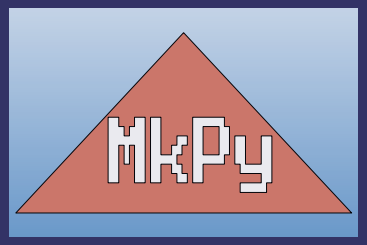


Schedules

- Final schedules

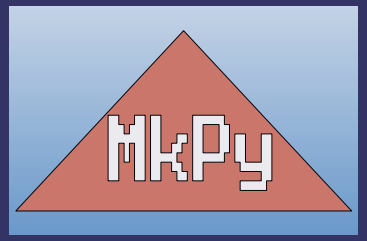






Next steps

- Implement mutations
- Prevent final schedules to be same as existing
- Find out how to save execution time
- Try some large scale tests (1 week/1month/0.5 year)
- Add more objectives (e.g. slew time, penalty for leaving high ranked observations out)
- Add constraints (BG, airmass, timing window, instrument)
- Make random obsPool more realistic (not fixed timeblocks)
- Add possibility to split long blocks
- Find real life example (transition from ‘fun’ to ‘useful’)



Credits

- Gemini Observatory for allowing me to take SSFA course
- Al Conrad (UHH) for pointing out this topic
- Petr Kubánek for demonstarting the feasibility
- Andrew Stephens (Gemini) for his twilight calculation
- Google/Wikipedia for info on Python/genetic algorithms
- GitHub: wreszelewski/ngsa2 (Python NGSA-II implementation)
- GitHub: matthewjwoodruff/pareto.py (Nondominated sorting)
- ....